IFW AF/2126

Docket No. AUS000153US1 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Taylor**	§	
	§	Group Art Unit: **2126**
Serial No.: **09/583,411**	§	
	§	Examiner: **Truong, Lechi**
Filed: **May 31, 2000**	§	
	§	
For: **Method and Apparatus for Bridging Service for Standard Object Identifier Based Protocols**	§	

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

ATTENTION: Board of Patent Appeals and Interferences

Certificate of Mailing Under 37 C.F.R. § 1.8(a)
I hereby certify this correspondence is being deposited with the United States Postal Service as First Class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on 08.10.04.

By: Amelia Turner
Amelia C. Turner

08/13/2004 SMINASS1 00000048 090447 09583411
01 FC:1402 330.00 DA

APPELLANT'S BRIEF (37 C.F.R. 1.192)

This brief is in furtherance of the Notice of Appeal, filed in this case on June 10, 2004.

The fees required under § 1.17(c), and any required petition for extension of time for filing this brief and fees therefore, are dealt with in the accompanying TRANSMITTAL OF APPEAL BRIEF.

This brief is transmitted in triplicate. (37 C.F.R. 1.192(a))

REAL PARTIES IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation

RELATED APPEALS AND INTERFERENCES

With respect to other appeals or interferences that will directly affect, or be directly affected by, or have a bearing on the Board's decision in the pending appeal, there are no such appeals or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

Claims in the application are: 1-57

B. STATUS OF ALL THE CLAIMS IN APPLICATION

1. Claims canceled: NONE
2. Claims withdrawn from consideration but not canceled: NONE
3. Claims pending: 1-57
4. Claims allowed: NONE
5. Claims rejected: 1-57

C. CLAIMS ON APPEAL

The claims on appeal are: 1-57

STATUS OF AMENDMENTS

The Advisory Action issued June 29, 2004, indicates that, for the purposes of appeal, the proposed amendments will be entered. However, no amendments were proposed after the Final Office Action issued March 10, 2004.

SUMMARY OF INVENTION

A method and apparatus is presented for maintaining a logical composite repository of Object Identifier (OID) tree structures on a server in a distributed data processing system. Each OID tree structure has been programmed to interface with an application programming interface (API) associated with an OID abstraction layer for the composite repository. See specification, page 8, lines 1-7. Any query from a requester in the distributed data processing system about an object contained in the logical composite repository associated with the server is processed by the OID abstraction layer. The query must be in a protocol, such as SNMP, LDAP, and CIM/XML, recognized by the OID abstraction layer. The repository associated with the object of the query is determined from the OID abstraction layer registry. The query is formatted to be consistent with the API associated with the OID abstraction layer and sent to the repository associated with the object. See specification, page 14, line 14, to page 15, line 6. The OID abstraction layer receives queries for objects in any of a number of different protocols. See page 8, lines 1-18. When a reply is received from the repository, it is formatted in the protocol of the original query and sent to the requester in the distributed data processing system.

ISSUES

The issues on appeal are as follows:

Whether claims 1-4, 20-23, 28, 35, 36, 39-42, 47, and 54-56 are unpatentable under 35 U.S.C. § 103(a) as being obvious over Spofford et al. (U.S. Patent No. 5,913,037) in view of Whitehead et al. (U.S. Patent No. 6,085,030) and Daigle (TISDAG--Monthly Report).

Whether claims 5-18, 24-27, 29-34, 37, 43-46, and 48-53 are unpatentable under 35 U.S.C. § 103(a) as being obvious over Spofford et al. in view of Whitehead et al. and Daigle and further in view of Furgeson (U.S. Patent No. 6,016,499).

Whether claims 29, 38, and 57 are unpatentable under 35 U.S.C. § 103(a) as being obvious over Spofford et al. in view of Whitehead et al. and Daigle and further in view of Applicant's allegedly admitted prior art.

GROUPING OF CLAIMS

The claims on appeal do not stand or fall in a single group, but are grouped into in the following groups for the reasons set forth in the Argument section below:

Claims 1-4, 20-23, and 39-42 form group A. Claims 28, 35, 36, 47, and 54-56 form group B. Claim 9 forms group C. Claims 5-8, 10-18, 24-27, 29-34, 37, 43-46, and 48-53 form group D. Claims 19, 38, and 57 form group E.

ARGUMENT

The Office Action rejects claims 1-4, 20-23, 28, 35-36, 39-42, 47, and 54-56 under 35 U.S.C. § 103 as being unpatentable over *Spofford et al.* (U.S. Patent No. 5,913,037) in view of *Whitehead et al.* (U.S. Patent No. 6,085,030) and further in view of *Daigle* (TISDAG -- Monthly report). This rejection is respectfully traversed.

I. The Applied References Cannot be Properly Combined to Form the Presently Claimed Invention (Groups A-E)

Spofford teaches a dynamic management information base manager. A management information base (MIB) manager allows agents to add or delete objects to any level within the MIB tree by object identifier (OID). See *Spofford*, Abstract. The MIB manager is a set of software interfaces, semantics, procedures, and data structures that work together to dynamically manage a tree of simple network management protocol (SNMP) data objects identified by an OID along with each object's value. SNMP is the only protocol contemplated by *Spofford*. *Spofford* does not teach or suggest an OID abstraction layer that is capable of receiving queries for objects in two or more different protocols, as recited in claim 1, for example.

The Advisory Action, issued June 29, 2004, states:

> As to the point (2), Spofford teaches the MIB manager 202, which executes the corresponding functions in response to the SNMP requests such as query [sic.] (col 11, 1-5). The SNMP requests that the present invention is applicable to any particular network protocol (col 1, ln 15-21). It is clearly show [sic.] that the MIB manager 202 has ability to received [sic.] information from two different protocols. The specification page 16, ln 5-10 indicated, "each repository must be programmed to work with this API, regardless of the protocol or protocol supported by the repository" which is more clearly to show the invention of application.

Appellant submits that the arguments presented in the Advisory Action clearly use the teachings in the instant application to make up for the acknowledged deficiencies in the applied references. The Final Office Action issued May 4, 2004, clearly states that *Spofford* and *Whitehead* do not teach "X" as queries for objects in two or more different protocols. The Examiner cannot use the present specification to bolster the teachings of the references.

Whitehead teaches a network component server that provides an object-neutral global component registry. See *Whitehead*, Abstract. Whitehead does not teach or suggest an OID abstraction layer that receives an OID tree structure from a repository and registers the tree structure with a registry associated with the OID abstraction layer, as recited in claim 1. More particularly, *Whitehead* does not teach or suggest an OID abstraction layer that is capable of receiving queries for objects in two or more different protocols, as recited in claim 1. Therefore, *Whitehead* does not make up for the deficiencies of *Spofford*.

Daigle teaches a technical infrastructure for Swedish directory access gateways (TISDAG). The TISDAG project taught by *Daigle* facilitates indexing of information from online Swedish directory service providers. Users may search the directory services for a person. The search results include full name of the person, e-mail address, organization, locality, full address, and telephone numbers. See *Daigle*, page 3 of the printout.

Non-analogous art cannot be used to establish obviousness. *In re Pagliaro*, 210 U.S.P.Q. 888, 892 (C.C.P.A. 1981). Although one of ordinary skill in the art is presumed to be aware of all prior art in the field to which the claimed invention pertains, he or she is not presumed to be aware of prior art outside of that field and the field of the problem to be solved. The presently claimed invention is directed towards the field of object identifiers (OIDs) for network resources, while *Daigle* is directed towards a whitepages directory for providing information about people. Therefore, *Daigle* is non-analogous art and cannot be used to form a *prima facie* case of obviousness.

Furthermore, *Spofford* and *Whitehead* teach object identifiers (OIDs) for components to be managed in a network. *Daigle* is not in the same field of endeavor as *Spofford* and *Whitehead*. Therefore, a person of ordinary skill in the art would not look to *Daigle* to solve the problems associated with *Spofford* and *Whitehead*. In fact, the Office Action does not recognize a problem associated with *Spofford* and/or *Whitehead* for which the teachings of *Daigle* would provide a solution. In fact, the Office Action alleges that a person of ordinary skill in the art

would find it obvious to combine *Spofford*, *Whitehead*, and *Daigle* with the motivation of "search refinement and retrieval of information." It is unclear how *Spofford*, *Whitehead*, and *Daigle* are being combined or how the proposed combination achieves the stated goal. Applicant asserts that a person of ordinary skill in the art would not be motivated to combine a whitepages directory gateway for retrieving information about people with an OID abstraction layer.

The Advisory Action issued June 29, 2004, states:

> As to the point (3), Daigle teaches capable [sic.] of receiving queries for objects in two or more different protocols, Daigle support the limitation that queries can come from different protocols (SNMP of Spofford)(page 5, sec: The DAG core).

However, the Examiner proffers no explanation of how a whitepages directory gateway is to be inserted into the management information base manager of *Spofford*, as combined with the network component server of *Whitehead*. Rather, the Examiner merely presents the pieces of the presently claimed invention in disparate teachings as if that is all that is required to establish a *prima facie* case of obviousness. In making an obviousness determination, one cannot pick and choose among the individual elements or assorted prior art references to recreate the claimed invention. *Symbol Technologies, Inc. v. Opticon, Inc.*, 935 F.2d 1569, 19 U.S.P.Q.2d 1241 (Fed. Cir. 1991). Instead, whether the prior art made obvious the invention must be determined by looking for some teaching or suggestion in the references to support their use in the particular claimed invention. *Id.*

Receiving queries in a plurality of protocols may be generally known; however, an OID abstraction layer that is capable of receiving queries in a plurality of protocols, particularly as recited and combined in claim 1, for example, is not shown in the prior art. Applying techniques used to locate people in a whitepages directory service in an OID abstraction layer would require undue experimentation and would amount to an inventive leap that can only be made with the benefit of Appellant's disclosure.

As shown above, the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation. Therefore, the proposed combination of *Spofford*, *Whitehead*, and *Daigle* does not render claim 1 obvious. Independent claims 20, 28, 39, and 47 recite subject matter addressed above with respect to claim 1 and are allowable for the same reasons. Since claims 2-4, 21-23, 35, 36, 40-42, and 54-56 depend from claims 1, 20, 28, 39, and 47, the same distinctions between *Spofford*, *Whitehead*, and *Daigle* and the invention recited in

claims 1, 20, 28, 39, and 47 apply for these claims. Additionally, claims 2-4, 21-23, 35, 36, 40-42, and 54-56 recite other additional combinations of features not suggested by the reference.

Therefore, Appellant respectfully requests that the rejection of claims 1-4, 20-23, 28, 35-36, 39-42, 47, and 54-56 under 35 U.S.C. § 103 not be sustained.

II. The Office Action Improperly Rejects Claims 28, 35, 36, 47, and 54-56 (Group B)

More particularly, with respect to claims 28, 35, 36, 47, and 54-56, the Office Action states:

> **As to claims 20-23, 28, 35, 36**, they are apparatus claims of claim 2, 3, 4, 9, 16, 17; they are rejected for the same reasons as claims 2, 3, 4, 9, 16, 17 above.
>
> **As to claims 39-42, 47, 54-56,** they are apparatus claims of claims 1-4, 9, 16-18; they are rejected for the same reasons as claims 1-4, 9, 16-18 above.

Office Action, dated March 10, 2004. Appellant respectfully disagrees. Claims 9 and 16-18 are rejected under different grounds of rejection. Clearly, corresponding claims 28, 35, 36, 47, and 54-56 are erroneously and improperly rejected. Appellant notified the Examiner of the improper rejection in the Response to Office Action, filed January 6, 2004, and in the Response to Final Office Action, filed May 4, 2004. However, the Examiner chose to maintain the improper rejection. Thus, Appellant respectfully requests that the rejection of claims 28, 35, 36, 47, and 54-56 not be sustained.

III. Ferguson Does Not Make Up for the Deficiencies of Spofford, Whitehead, and Daigle (Groups C and D)

The Office Action rejects claims 5-18, 24-27, 29-34, 37, 43-46, and 48-53 under 35 U.S.C. § 103 as being unpatentable over *Spofford* in view of *Whitehead* and *Daigle* and further in view of *Ferguson* (U.S. Patent No. 6,016,499). This rejection is respectfully traversed.

With respect to independent claim 9, the Final Office Action states:

> **As to claim 9**, Spofford teaches a first query (a query, col 10, ln 25-67 to col 11, ln 1-16), the object data (the objects, col 10, ln 25-67 to col 11, ln 1-16), a request (request, col 10, ln 25-67 to col 11, ln 1-16), a protocol (SNMP, col 1, ln 1-35/ protocol, col 5, ln 5-67/ col 6, ln 1-67), OID (OID, col 2, ln 59-67, col 6, ln 1-45, col 4, ln 1-9, col 7, ln 20-62, col 8, ln 15-52), abstraction layer (MIB manager, (OID, col 2, ln 59-67/ col 6, ln 1-45/ col 4, ln 1-9/ col 7, ln 20-62/col 8, ln 15-52/ col 11, ln 1-30/ col 12, ln 40-67), API.

Office Action, dated March 10, 2004. Appellant agrees that *Spofford* teaches a query, object data, a request, a protocol, and an MIB manager. *Spofford* does not make any mention whatsoever of an API. However, independent claim 9 does not merely recite a query, object data, a request, a protocol, an MIB manager, and an API. Rather, claim 9 recites the following:

> 9. A method on a server in a distributed data processing system for retrieving object data from a repository, comprising:
>
> receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols;
>
> locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and
>
> retrieving the object data from the repository using an OID abstraction layer application program interface (API).

The Office Action does not address the specific features recited in claim 9 other than to simply list some features from one of the references. More particularly, the Office Action does not address a step of receiving a first query for object data in a protocol recognized by an OID abstraction layer, wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols. The Office Action does not even attempt to establish a *prima facie* case of obviousness for claim 9. The Office Action fails to address the features that *Spofford* does not teach or how Whitehead, *Daigle*, and *Ferguson* allegedly make up for the deficiencies of *Spofford* with respect to claim 9.

Ferguson does generally teach application program interfaces (API) and, more specifically, teaches an API that includes at least one callable element that is capable of accessing a component of a repository in response to being called and a driver that is capable of translating a database language statement, such as an SQL statement, into an executable API sequence. However, *Ferguson* does not teach or suggest an OID abstraction layer that is capable of receiving queries for objects in two or more different protocols, as recited in claim 1. Therefore, *Ferguson* does not make up for the deficiencies of *Spofford*, *Whitehead*, and *Daigle*. As such, the proposed combination of *Spofford*, *Whitehead*, *Daigle*, and *Ferguson* cannot render obvious the present invention, as further limited by claims 5-8, 10-18, 24-27, 29-34, 37, 43-46, and 48-53. Since the applied references, taken alone or in combination, fail to teach or suggest each and every claim limitation, claims 5-8, 10-18, 24-27, 29-34, 37, 43-46, and 48-53 are not rendered

obvious by the proposed combination of *Spofford*, *Whitehead*, *Daigle*, and *Ferguson*.

Therefore, Applicant respectfully requests that the rejection of claims 5-18, 24-27, 29-34, 37, 43-46, and 48-53 under 35 U.S.C. § 103 not be sustained.

IV. The Allegedly Admitted Prior Art Does Not Make Up for the Deficiencies of Spofford, Whitehead, and Daigle (Group E)

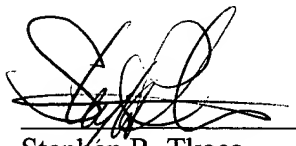
The Office Action rejects claims 19, 38, 57 under 35 U.S.C. § 103 as being unpatentable over *Spofford* in view of *Whitehead* and *Daigle* and further in view of allegedly admitted prior art (APA). This rejection is respectfully traversed.

There is no suggestion or motivation whatsoever in *Spofford* for using CIM/XML. The MIB manager of *Spofford* is incapable of processing queries in CIM/XML protocol. The mere fact that a prior art reference can be readily modified does not make the modification obvious unless the prior art suggested the desirability of the modification. *In re Laskowski*, 871 F.2d 115, 10 U.S.P.Q.2d 1397 (Fed. Cir. 1989) and also *see In re Fritch*, 972 F.2d 1260, 23 U.S.P.Q.2d 1780 (Fed. Cir. 1992) and *In re Mills*, 916 F.2d 680, 16 U.S.P.Q.2d 1430 (Fed. Cir. 1993). The Office Action may not merely state that the modification would have been obvious to one of ordinary skill in the art without pointing out in the prior art a suggestion of the desirability of the proposed modification. In this case, the only suggestion or motivation for making the proposed modification is found in Appellant's own specification. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed modification, the presently claimed invention can be reached only through the an impermissible use of hindsight with the benefit of Appellant's disclosure a model for the needed changes.

Claims 38 and 57 recite subject matter addressed above with respect to claim 19 and are allowable for the same reasons. Therefore, Appellant respectfully requests that the rejection of claims 19, 38, and 57 under 35 U.S.C. § 103 not be sustained.

V. Conclusion

In view of the above, Appellant respectfully submits that the rejections of claims 1-57 are overcome. Accordingly, it is respectfully urged that the rejections of claims 1-57 not be sustained.



Stephen R. Tkacs
Reg. No. 46,430
YEE & ASSOCIATES, P.C.
PO Box 802333
Dallas, TX 75380
(972) 367-2001

APPENDIX OF CLAIMS

The text of the claims involved in the appeal reads:

1. A method on a server in a distributed data processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the method comprising the steps of:

receiving, in an OID abstraction layer, an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols;

registering the OID tree structure with a registry associated with the OID abstraction layer; and

adding the OID tree structure to a repository associated with the OID abstraction layer.

2. The method of claim 1, wherein the registry associated with the OID abstraction layer provides information identifying an anchor point in the OID subtree structure to be maintained by the repository.

3. The method of claim 2, wherein if the anchor point of the OID subtree structure is already registered with the OID abstraction layer, the registry is overwritten.

4. The method of claim 2, wherein if a query is received for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure, the OID abstraction

layer identifies a repository that maintains object information for the requested object based on the registered anchor point.

5. The method of claim 1, wherein the repository is configured such that the repository recognizes requests from an application program interface (API) associated with the OID abstraction layer and sends reply messages to the API containing information retrieved from the repository.

6. The method of claim 5, wherein the OID abstraction layer receives the information retrieved from the repository through the API and encapsulates the information in a reply message to a protocol interface.

7. The method of claim 1, wherein the OID abstraction layer receives a request for object data from a protocol interface, converts the request into an application program interface (API) request which is forwarded to the repository, and receives an API reply from the repository having the object data.

8. The method of claim 7, wherein the OID abstraction layer reformats the object data in a reply message and sends the reply message to the protocol interface.

9. A method on a server in a distributed data processing system for retrieving object data from a repository, comprising:

receiving a first query for the object data from a requester in the distributed data

processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols;

locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

retrieving the object data from the repository using an OID abstraction layer application program interface (API).

10. The method of claim 9, wherein the first query is mapped into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

11. The method of claim 10, wherein if the first query cannot be mapped into a second query due to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.

12. The method of claim 10, wherein the second query is sent to the repository that contains the object associated with the first query.

13. The method of claim 12, wherein a first reply is received at the API associated with the OID abstraction layer from the repository that contains the object associated with the first query.

14. The method of claim 13, wherein the first reply is transformed into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer.

15. The method of claim 14, wherein the second reply is sent to the requester in the distributed data processing system.

16. The method of claim 9, wherein each repository in the plurality of repositories contains information representing an Object Identifier (OID) subtree structure.

17. The method of claim 9, wherein Simple Network Management Protocol (SNMP) is a protocol recognized by the OID abstraction layer.

18. The method of claim 9, wherein Lightweight Directory Access Protocol (LDAP) is a protocol recognized by the OID abstraction layer.

19. The method of claim 9, wherein Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) is a protocol recognized by the OID abstraction layer.

20. An apparatus on a server in a distributed data processing system for maintaining a logical composite repository of Object Identifier (OID) tree structures, the apparatus comprising:

an OID abstraction layer that receives an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different

protocols;

a registry, associated with the OID abstraction layer, that registers the OID tree structure; and

an adding means for adding the OID tree structure to a repository associated with the OID abstraction layer.

21. The apparatus of claim 20, wherein the registry provides information identifying an anchor point in the OID tree structure to be maintained by the repository.

22. The apparatus of claim 21, wherein if the anchor point of the OID tree structure is already registered in the registry, then the registry overwrites the previous entry.

23. The apparatus of claim 21, wherein, if the OID abstraction layer receives a query for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure, the registry in the OID abstraction layer identifies a repository that maintains object information for the requested object based on the registered anchor point.

24. The apparatus of claim 20, wherein the repository is configured such that the repository recognizes requests received from an application program interface (API) associated with the OID abstraction layer and sends reply messages to the API containing information retrieved from the repository.

25. The apparatus of claim 24, wherein the OID abstraction layer receives the information retrieved from the repositories through the API and encapsulates the information in a reply message to a protocol interface.

26. The apparatus of claim 20, wherein the OID abstraction layer receives a request for object data from a protocol interface, converts the request into an application program interface (API) request which is forwarded to the repository, and receives an API reply from the repository having the object data.

27. The apparatus of claim 26, wherein the OID abstraction layer encapsulates the object data in a reply message and sends the reply message to the protocol interface.

28. An apparatus on a server in a distributed data processing system for retrieving object data from a repository, comprising:

a receiving means for receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols;

a locating means for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

a retrieving means for retrieving the object data from the repository using an OID abstraction layer application program interface (API).

29. The apparatus of claim 28, further comprising a mapping means for mapping the first query into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

30. The apparatus of claim 29, wherein if the mapping means cannot map the first query into a second query due to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.

31. The apparatus of claim 29, further comprising a first sending means, in the OID abstraction layer, that sends the second query to a repository that contains the object associated with the first query.

32. The apparatus of claim 31, wherein the retrieving means receives a first reply at the API from the repository that contains the object associated with the first query.

33. The apparatus of claim 32, further comprising a transforming means, in the OID abstraction layer, that transforms the first reply into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer.

34. The apparatus of claim 33, further comprising a second sending means, in the OID abstraction layer, that sends the second reply to the requester in the distributed data processing system.

35. The apparatus of claim 28, wherein each repository contains Object Identifier (OID) tree structures.

36. The apparatus of claim 28, wherein the receiving means recognizes a Simple Network Management Protocol (SNMP) query.

37. The apparatus of claim 28, wherein the receiving means recognizes a Lightweight Directory Access Protocol (LDAP) query.

38. The apparatus of claim 28, wherein the receiving means recognizes a Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) query.

39. A computer program product in a computer readable medium for maintaining a repository of Object Identifier (OID) tree structures, comprising:

instructions for receiving, in an OID abstraction layer, an OID tree structure from a repository; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols;

instructions for registering the OID tree structure with a registry associated with the OID abstraction layer; and

instructions for adding the OID tree structure to a repository associated with the OID abstraction layer.

40. The computer program product of claim 39, further comprising instructions for maintaining the registry associated with the OID abstraction layer and providing information identifying an anchor point in the OID tree structure to be maintained by the repository.

41. The computer program product of claim 40, wherein if the anchor point of the OID tree structure is already registered with the OID abstraction layer, the instructions for registering overwrites the previous entry.

42. The computer program product of claim 40, further comprising instructions for identifying a repository that maintains object information for the requested object based on the registered anchor point if a query is received for an object that has an Object Identifier that is below a registered anchor point in an OID tree structure.

43. The computer program product of claim 39, further comprising instructions for configuring the repository to recognize requests from an application program interface (API) associated with the OID abstraction layer and to send reply messages to the API containing information retrieved from the repository.

44. The computer program product of claim 43, further comprising instructions for receiving the information retrieved from the repository, through the API, and encapsulating the information in a reply message to a protocol interface.

45. The computer program product of claim 39, further comprising instructions for receiving a request for object data from a protocol interface;

instructions for converting the request into an application program interface (API) request which is forwarded to the subtree repository; and

instructions for receiving an API reply from the subtree repository having the object data.

46. The computer program product of claim 45, further comprising instructions for encapsulating the object data in a reply message and sending the reply message to the protocol interface.

47. A computer program product in a computer readable medium for retrieving object data from a repository, comprising:

instructions for receiving a first query for the object data from a requester in the distributed data processing system, wherein the first query is in a protocol recognized by an OID abstraction layer; wherein the OID abstraction layer is capable of receiving queries for objects in two or more different protocols;

instructions for locating a repository that contains the object data requested in the first query based on a registry associated with the OID abstraction layer; and

instructions for retrieving the object data from the repository using an OID abstraction layer application program interface (API).

48. The computer program product of claim 47, wherein the instructions for receiving the first query map the first query into a second query, wherein the second query is consistent with an application program interface (API) associated with the OID abstraction layer.

49. The computer program product of claim 48, wherein if the instructions for receiving the first query map cannot map the first query into a second query due to a limitation of the repository that contains the object associated with the first query, then the first query cannot be satisfied.

50. The computer program product of claim 48, further comprising instructions for sending the second query to the repository that contains the object associated with the first query.

51. The computer program product of claim 50, further comprising instructions for receiving a first reply at the API associated with the OID abstraction layer from the repository that contains the object associated with the first query.

52. The computer program product of claim 51, further comprising instructions for transforming the first reply into a second reply, wherein the second reply is consistent with the protocol for the first query recognized by the OID abstraction layer.

53. The computer program product of claim 52, further comprising instructions for sending the second reply to the requester in the distributed data processing system.

54. The computer program product of claim 47, wherein the repository contains Object Identifier (OID) tree structures.

55. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Simple Network Management Protocol (SNMP) query.

56. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Lightweight Directory Access Protocol (LDAP) query.

57. The computer program product of claim 47, wherein instructions for receiving a first query recognize a Common Information Model used in conjunction with eXtendable Markup Language (CIM/XML) query.